Lab 6

Levon Demirdjian

Tuesday, May 03, 2016

Importance sampling: Example 1

Say we want to approximate the mean of a standard normal distribution that is truncated to the unit interval $[0, 1]$. The density of such a random variable is given by

$$f(x) = \frac{\phi(x)}{\int_0^1 \phi(x)dx}, \tag{1}$$

where $\phi(x)$ is the pdf of a $N(0, 1)$ random variable, i.e.

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp(-0.5 * x^2).$$

It should be clear that $f(x)$ is a density (nonnegative and integrates to 1).

Problem 1
Implement importance sampling using $g(x) \sim$ unif[0,1] to approximate $E_f[X]$.

Solution 1
Let $X_1, ..., X_n \overset{iid}{\sim} unif[0, 1]$. Then

$$\hat{I} = \frac{1}{n}\sum_{i=1}^{n} \frac{f(X_i)}{g(X_i)} X_i \to E_f[X],$$

so the left hand side is our importance sampling estimator of the expectation of X (under f).

```
truncated_density <- function(x){
  dnorm(x) / (pnorm(1) - pnorm(0))
}

importance_weight <- function(x){
  truncated_density(x) / 1
  ## Dividing by the density of unif(0,1), i.e. 1
}

n <- 10000
Y <- runif(n)
mean(importance_weight(Y) * Y)
```

```
## [1] 0.4565722
```

Importance sampling: Example 2

Let $Z \sim N(0, 1)$. We would like to estimate the tail probability $P(Z > c)$, where c is a large constant, for example $c = 8$. There are several ways to approach this problem using Monte Carlo methods.

First of all, we can compute the exact value of this probability so we can measure the quality of our estimation. This probability is given by:

```r
pnorm(8, lower.tail = FALSE)
```

```
## [1] 6.220961e-16
```

We see that the true probability is given by

$$P(Z > c) = 6.220961 * 10^{-16} \tag{2}$$

(this is an incredibly small number). Let's consider the usual Monte Carlo setup.

Let $Z_1, ..., Z_n \overset{iid}{\sim} N(0, 1)$ and construct the estimator $\hat{I}$, where

$$\hat{I} = \frac{1}{n}\sum_{i=1}^{n} 1_{Z_i > c} \approx E(1_{Z>c}) = P(Z > c). \tag{3}$$

That is, we expect $\hat{I}$ to be a reasonable estimator of $P(Z > c)$ since the LLN implies that the former converges to the latter for large sample sizes. Let's code this up:

```r
set.seed(14)
n <- 1000
c <- 8
Z <- rnorm(n)
mean(Z > c)
```

```
## [1] 0
```

We see that our estimate is way off, since none of the sampled points were larger than 8. What if we use a larger n?

```r
set.seed(14)
n <- 100000
c <- 8
Z <- rnorm(n)
mean(Z > c)
```

```
## [1] 0
```

Still 0! This event is incredibly rare, that in order to see it we need really, really large sample sizes. This can be seen by evaluating the variance of the estimator $\hat{I}$:

Problem 2
Compute the variance of $\hat{I}$.

Solution 2
$Var(\hat{I}) = \frac{1}{n}Var(1_{Z_i>c}) = \frac{1}{n}P(Z > c)[1 - P(Z > c)] = 6.220961 * 10^{-20}$ for $n = 10,000$.
Note this variance may seem small, but it is very large! The quantity we are interested in is $6.220961 * 10^{-16}$ and the sd of our estimator is $2.494185 * 10^{-10}$.
Therefore, let's consider a method based on importance sampling to remedy this issue.

We are going to simulate variables $Y_1, .., Y_n$ from an alternative distribution $g(y)$, notably the shifted exponential distribution with pdf

$$g(y) = \exp^{-(y-c)} 1_{y>c}. \tag{4}$$

It's easy to simulate from $g(y)$: simply simulate from a regular exponential(1) and add c to the simulated value.

Problem 3
Write down the form of the importance sampling estimator of $\tilde{I}$.

Solution 3

$$\tilde{I} = \frac{1}{n}\sum_{i=1}^{n} \frac{f(Y_i)}{g(Y_i)} 1_{Y_i>c} = \frac{1}{n}\sum_{i=1}^{n} 1_{Y_i>c} \frac{1}{\sqrt{2\pi}} \exp(-0.5 * Y_i^2 + Y_i - c).$$

Problem 4
Compute the variance of $\tilde{I}$ (hint: use the fact that $Var(h(X)) = E(h(X)^2) - E(h(X))^2$). You will also need to use the integrate function in R to compute an integral. How does it compare to the variance of $\hat{I}$?

Solution 4

$$\begin{aligned}
Var(\tilde{I}) &= \frac{1}{n} Var_g\left[\frac{f(Y)}{g(Y)} 1_{Y>c}\right] = \frac{1}{n}\left(E_g\left[\frac{f^2(Y)}{g^2(Y)} 1_{Y>c}\right] - \left[E_g\left(\frac{f(Y)}{g(Y)} 1_{Y>c}\right)\right]^2\right) \\
&= \frac{1}{n}\left[\int_c^\infty \frac{f^2(y)}{g(y)} dy - P(Z>c)^2\right] \\
&= 1.299953 * 10^{-35}.
\end{aligned}$$

Code this up:

```
thing_in_integral <- function(x, c){
  (dnorm(x)^2) / exp(c-x)
}
tail_prob <- pnorm(c, lower.tail = FALSE)

(integrate(thing_in_integral, lower = c, upper = Inf, c = c)$value - tail_prob^2)/n
```

```
## [1] 1.299953e-35
```

So the sd of this method is $3.605486 * 10^{-18}$ for $n = 10,000$. Compare this to the sd of the regular MC method, $2.494185 * 10^{-10}$. The performance has been improved by roughly a factor of 10^8. Wow!

Problem 5
Use this importance sampling scheme to approximate the quantity of interest, $P(Z > c)$. I'm hiding my code (try to do this by yourself)!

```
## [1] 6.190064e-16
```